NEXT STOP: TERAFLOP; THE PARALLELIZATION OF AN ATMOSPHERIC GENERAL CIRCULATION MODEL

Daniel S. Schaffer* and Max J. Suárez

NASA Seasonal to Interannual Prediction Project
NASA Goddard Space Flight Center
Code 971, Building 22, Greenbelt, MD 20771
dans@janus.gsfc.nasa.gov

Keywords: parallel, atmospheric, gcm

ABSTRACT

In the 1990's, computer manufacturers are increasingly turning to the development of parallel processor machines to meet the high performance needs of their customers. Simultaneously, atmospheric scientists studying weather and climate phenomena ranging from hurricanes to El Niño to global warming require increasingly fine resolution models. Here, implementation of a parallel atmospheric general circulation model (GCM) which exploits the power of massively parallel machines is described. Using the horizontal data domain decomposition methodology, this FORTRAN 90 model is able to integrate a 0.6° longitude by 0.5° latitude problem at a rate of 19 Gigaflops on 512 processors of a Cray T3E 600; corresponding to 280 seconds of wall-clock time per simulated model day. At this resolution, the model has 64 times as many degrees of freedom and performs 400 times as many floating point operations as the model it replaces.

INTRODUCTION

The general circulation modeling community constantly demands more computing power to meet its needs. Short to medium range weather forecasters have used increasingly faster machines to run higher resolution models. The improved solutions obtained from higher resolution in numerical weather prediction is well known; Simmons, et al. (1989), among others, document this. Higher resolution is also important to seasonal and interannual variability studies (e.g. Déqué and Piedlievre, 1995 and Lal, et al., 1997). For studies of longer time scale phenomena, completing model runs at any reasonable resolution becomes the challenge. Coupled atmospheric/ocean simulations of El Niño require enormous computational power. Recently, some modelers have turned to ensembles of runs to produce better predictions; an exercise that magnifies resource demands. For the time scales of global climate change, coupled model runs can last hundreds of simulated years (e.g. Manabe and Stouffer, 1994); for studies of the thermohaline circulation, those numbers stretch into the thousands.

*SAIC General Sciences Corporation; Laurel, MD

To meet these needs, super-computer manufacturers have offered a variety of solutions. Since the 1980's, parallel vector processors have been the most widely used by the GCM community. However, in the 1990's cache-based massively parallel processor (MPP) machines have become increasingly prominent. These machines present a dual challenge to model designers of writing code that runs efficiently within a single processor yet scales nearly linearly for hundreds, maybe thousands of processors.

A snapshot of the progress of (mostly atmospheric) model designers toward meeting these challenges was presented in a special issue of Parallel Computing in 1995. Drake, et al. (1995) wrote a message passing implementation of the National Center for Atmospheric Research (NCAR) Community Climate Model (CCM2) for the IBM SP2 and Intel Paragon machines. Most notable was the poor single processor performance they attributed to inefficient cache use (a result noted repeatedly in the literature). Jones, et al. (1995) implemented a parallel version of the Geophysical Fluid Dynamical Laboratory (GFDL) Atmospheric General Circulation Model (AGCM) running on the the CM-5 and SGI/Cray C90. Single processing element (PE) performance and scaling were quite good on the C90 but hampered on the CM5 by over-use of memory they attributed to poor algorithmic design. Lou and Farrara (1996) optimized a parallel version of the UCLA AGCM for the Paragon and SGI/Cray T3D/E. The model scales fairly well but their preliminary attempts at cache-based optimizations have yielded modest improvements.

Here we describe the parallel design and performance of the NASA Goddard Space Flight Center (GSFC) atmospheric model. This model will be the production version used in the NASA Seasonal to Interannual Prediction Project (NSIPP). The primary objectives are efficient single PE performance and scalability on MPPs. Section 2 describes the scientific basis of the model. Section 3 explains the high-level model design, the parallelization methodology, and gives highlights of the detailed design. Section 4 analyzes the model performance. Section 5 discusses future directions for this effort. The model is similar in structure to a stratospheric version under development by the Data Assimilation Office at NASA/GSFC and described in a companion paper (Sawyer, et al., submitted to HPC '98).

MODEL DESCRIPTION

The dynamical portion of the GCM is based on a finite-differenced, primitive equations dynamical core (Dycore) (Suarez and Takacs, 1995) that allows arbitrary horizontal and vertical resolution. Its prognostic variables are the two horizontal wind components, the potential temperature, the surface pressure, the water vapor mixing ratio, and an arbitrary number of passive tracers. In the vertical, a discretization scheme is used which closely follows that proposed by Arakawa and Suarez (1983), but applied to a generalized vertical coordinate. In the horizontal, the equations are finite-differenced on a staggered latitude-longitude grid (the C-grid). To avoid linear computational instability due to meridional convergence, a Fourier filter is applied to prognostic variable tendencies poleward of 45 degrees latitude. The model also filters the prognostic variables (Shapiro, 1970) to damp small-scale waves that can lead to non-linear computational instability. The model is integrated in time using a leapfrog scheme modified as proposed by Brown and Campana (1978) and by applying a weak time filter (Asselin, 1972).

The solar parameterization (Chou, 1992) models absorption due to O_3, CO_2, water vapor, O_2, clouds, and aerosols, as well as gaseous, cloud, and aerosol scattering. The infrared parameterization (Chou and Suarez, 1994) includes absorption by water vapor, CO_2, O_3, methane, N_2O, CFC-11, CFC-12 and CFC-22 within eight spectral bands. Other parameterizations include the Louis et al. (1982) turbulence and Zhou et al. (1996) gravity wave drag schemes. Penetrative convection originating in the boundary layer is modeled using the Relaxed Arakawa-Schubert (RAS) scheme (Moorthi and Suarez, 1992). The Mosaic land surface model (LSM) (Koster and Suarez, 1992) computes area-averaged energy and water fluxes from the land surface in response to meteorological forcing. A grid square is sub-divided into relatively homogeneous sub-regions ("tiles" of the mosaic), each containing a single vegetation or bare soil type.

COMPUTATIONAL DESIGN

We begin by describing the high level structure of the GCM so as to provide context for the results in section 4. The model is divided into self-contained components, each operating on its own space (grid) and time scales. "Coupling" software converts data from one model grid to another in parallel. (The couplers are analogous to the NCAR Climate System Model (CSM) flux coupler; Bryan, et al., 1996). The GCM driver that ties together these components can be atmospheric only, ocean only, coupled atmospheric/ocean, etc. Presently, the major components for this AGCM are:

1. Dynamics - Dycore, the Shapiro filter and the model stepping functionality.

2. Slow Physics - The longwave and shortwave radiation calculations.

3. Fast Physics - The remainder of the AGCM; convection, turbulence, land processes, etc.

The parallelization is implemented using a horizontal data domain decomposition. Put simply, each processor operates on a slab of data extending from the surface to the top of the atmosphere. The primary advantage of this decomposition is that the number of points available to split among the processors is large; allowing utilization of hundreds or thousands of PE's. In addition, physics calculations such as longwave, shortwave, etc. become "embarrassingly parallel". Finally, at a practical level, using this scheme means that the original plug compatible physics subroutines can be retained, unmodified, in the parallel implementation.

The processors are laid out in a rectangular array so that each PE has exactly one neighbor on each of four sides. The number of PE's in the X and Y direction (NX, NY) as well as the number of grid points within a PE (IM, JM) are arbitrary. Ghost (shadow) regions are defined to facilitate nearest neighbor communication. When code such as horizontal advection needs to access an array element outside the processor bounds, a communications call is made to fill in the ghost region. Once the data are in place, the code can iterate to its full extent as if it were written for a serial model. The communication is bundled over levels to reduce the impact of latency.

Since the primary objective is implementation on a distributed memory MPP, a message-passing scheme is used for the communication. Generic synchronous point to point send/receive routines provide the backbone for this scheme. Currently they are implemented using calls to either native Cray shared memory software (SHMEM) or message passing interface (MPI) routines. This backbone is packaged into a single "communication primitives" module. Since this is the only code that varies between implementations, porting the model is quite simple.

While most of the communication in the model is nearest neighbor, the polar filter is a significant exception. It is implemented by first transposing the data from an (X,Y) to a (Y,Z) decomposition, then executing local FFTs, then transposing back. This implies that the greater the decomposition in X, the poorer the performance of the polar filter. Conversely, nearest neighbor communication scales as $\frac{1}{\sqrt{PEs}}$ only if the processor layout is close to symmetrical. These conflicting performance considerations guide optimal processor layout choice and represent the most obvious disadvantage of this decomposition strategy.

Currently, no load balancing is implemented. The sources of imbalance are as follows: 1. The shortwave code; radiative transfer calculations need only be performed for sunlit soundings. 2. The land surface code; no computations are needed for ocean points and the uneven distribution of tiles further un-balances the problem. 3. Cumulus convection; fewer computations are needed where convection does not occur. 4. The polar filter; it only operates poleward of 45 degrees latitude. The utility of implementing load balancing schemes will be discussed in section 5.

All new code is written in FORTRAN 90. Array syntax, user-defined types, subroutine overloading, modules, and dynamic memory allocation are used extensively. Use of these features has helped to create reasonably well-structured code and greatly facilitated debugging. Since all memory is dynamically allocated, the model runs at any resolution using any processor layout without recompilation. On the downside, dynamic memory use may hamper future cache-based optimizations.

RESULTS AND PERFORMANCE

The model is currently being run on the DEC Alpha workstation, Cray T3E, and Cray J90. To validate the code, results from the original production version and the new GCM were compared for the same initial and boundary conditions at a resolution of 72x45x22. At this resolution, Dynamics and Fast Physics run at 9 minute intervals; Slow Physics every 3 hours. After 3 hours, checksums of state variables, budgets and other diagnostic quantities for the old and new code differ at the round-off level; for one or multiple processors.

To assess performance, the floating point operations (FLOPs) are counted for a one processor run using the J90 PERF utility. Initialization and finalization times are not counted. No model output is done during the "run" phase for purposes of these benchmarks. Performance is then computed by dividing the FLOP count by the run-time measured by wall-clock timers. The 72x45x22 resolution problem was run on the Cray T3E-600 using 32 bit words for up to 64 PEs. The peak performance is 1.35 Gflop/s, corresponding to 20 seconds run-time per simulated model day. A 64 bit version runs at only 28 seconds per day; largely due to the fact that the code is memory-access bound. In comparison, the original production version running multi-tasked on the Cray J90 (64 bit) simulates one model day in 50 seconds.

To truly exploit the power of the T3E machine, we turn to a high resolution problem; 576x360x22 (0.625° by 0.5° by 22 levels). Preliminary tests show a Dynamics time step of one minute is required to satisfy the Courant-Friedrich-Levy (CFL) condition for linear numerical stability at this resolution. The Fast Physics is run every 10 minutes and Slow Physics at 3 hour intervals. For a 3 hour run, the floating point operations total 686 billion. The 32 bit version requires approximately 1 billion words of memory; translating to a minimum of 64 Cray T3E-600 PEs. The GCM was tested for processor configurations totaling up to 512 PEs. Experimentation showed that for 512 PEs, a processor layout of 16 PEs in longitude, 32 in latitude is optimal. For that case, the performance is 19.6 Gflop/s. This corresponds to 280 seconds of wall-clock time per simulated model day.

The details of the T3E performance are shown in the speedup plots in figure 1. The solid lines in the figure are curve fits of the data to Amdahl's speedup law:

$$S = \frac{1}{Fs + \frac{Fp}{NP}}$$

where S is the speedup, Fs is the serial fraction, Fp is the parallel fraction and NP is the number of processors. For a perfectly load balanced code, the effective single pe performance is an estimate of how fast it would run on 1 PE if that were possible. Notice that, in Dynamics, this number is higher than the per-processor performance because it does not include the degradation due to communication as the problem is scaled to 512 PEs. The floating point operation counts show that Dynamics is responsible for the great majority of the work. This is largely due to its relatively short time steps. The fact that Slow Physics does not scale perfectly is currently under investigation.

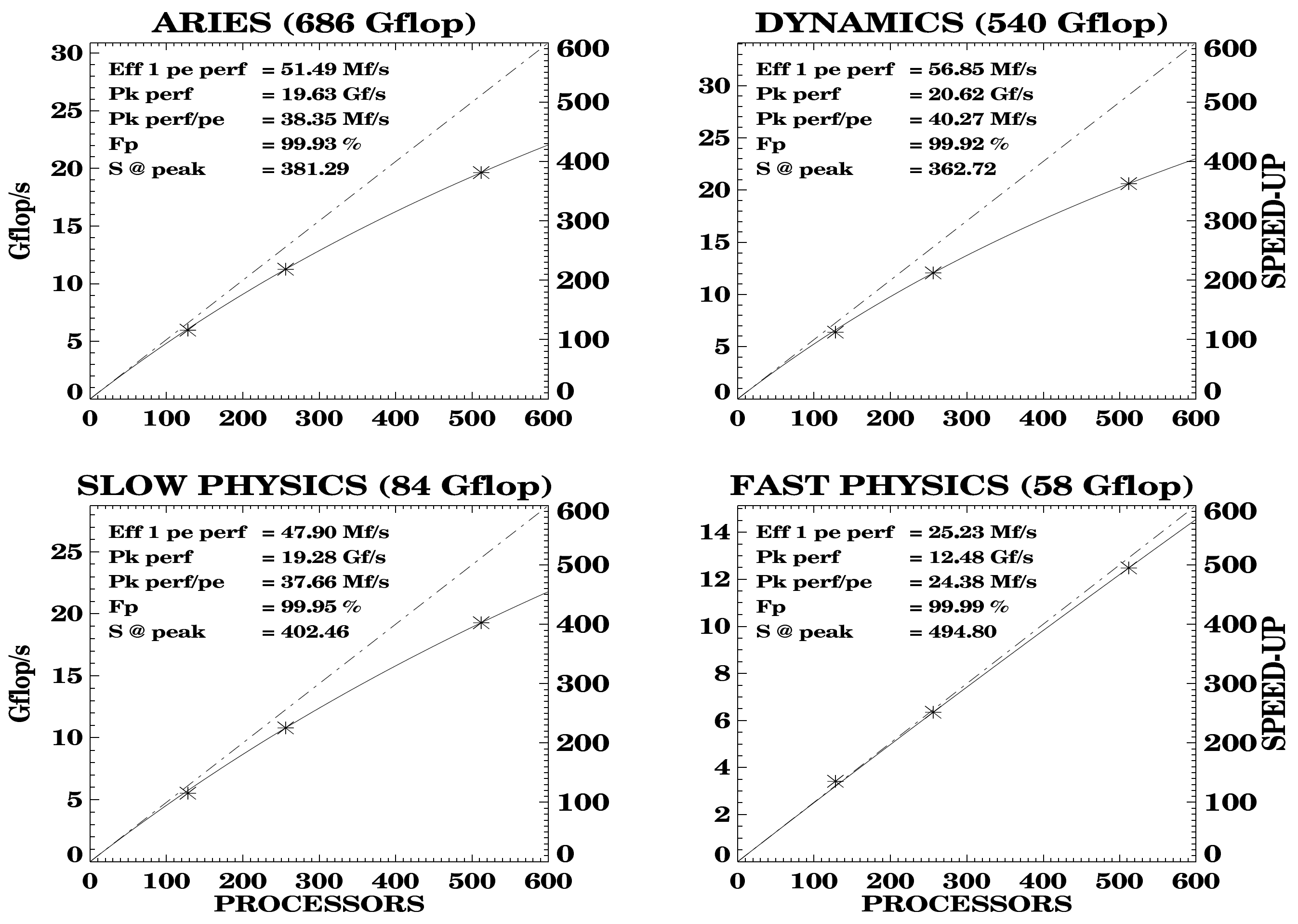

Figure 1: Speedup plots for 3 hour runs of the full GCM and its three major components. The floating point operations in billions are given at the top of each graph. The asterisks represent the speed in Gflop/s for 128, 256 and 512 PEs. The dot-dashed line represents a perfectly linear speedup. The solid curve was obtained by fitting the operations and run-times to Amdahl's speedup law (see text). Fp and S are as given in Amdahl's law. The effective single PE performance is the curve value for NP=1.

Code	GFLOP	Time (s)	Gf	Mf/PE
Dycore	427.3	17.77	24.1	47.0
Shapiro	74.6	5.56	13.4	26.2
Step	33.6	1.57	21.4	41.8
Longwave	34.3	0.97	35.4	69.1
Shortwave	48.0	2.77	17.3	33.8
LSM	6.8	0.96	7.1	13.8
RAS	4.6	1.21	3.8	7.4

Table 1: Floating point operations (in billions), run-time, total performance, and per pe performance for a 3 hour run of the 576x360x22 resolution problem at 512 PEs.

Table 1 shows a breakdown of performance of the major GCM pieces. The dynamical core consumes most run-time and will need the greatest attention during future optimizations. The poor scaling of the Shapiro filter is expected; it does relatively few floating point operations per communication. That the Step function does not scale perfectly is merely an artifact of the code design. It fills the ghost regions of the state variables; work that could just have easily been done in Dycore. The LSM and RAS codes are "super-scalar". This commonly observed result occurs because as the number of processors increases, the amount of memory needed per pe decreases and, consequently, the data fit better in cache.

The rated performance of the Cray T3E 600 is 600 Mflop/s. While, in practice, few codes reach 200 Mflop/s per PE, it is clear from table 1 that single-pe performance is sub-par. One reason is poor cache re-use. As a first cut, this code was written to mimic the original serial code which was designed to run efficiently on vector machines. As of yet, no serious cache-based optimizations have been attempted. A second reason is communication costs. Measurements by the T3E Apprentice utility indicate that 40% of the Dycore run-time is communication. Latency is significant. Even with bundled Ghost calls, preliminary measurements indicate that 30% of the nearest neighbor communication time is latency. When the Ghost calls are unbundled, Dycore performance degrades by 20%. A third cause of the poor single-pe performance is load-imbalance as described earlier.

The same resolution running on one PE of a Cray J90 performed at 90 Mflop/s. Since the rated performance of the J90 is 200 Mflop/s, the model is clearly vectorizing quite well. Although a multiple PE version has not been run for this resolution, past experience suggests that it should perform at about 1 Gflop/s for 16 PEs. An MPI implementation on the J90 was found to significantly degrade the code's performance; presumably due to the high level of overhead in the MPI software. A T3E MPI version has not been tested.

FUTURE DIRECTIONS AND SUMMARY

As mentioned, the code produces the correct answer for a low resolution problem in which Dynamics and Fast Physics are called at the same frequency. For the high resolution case, the code runs but does not produce the correct answer. The first step will be to correct these errors and assess the climatology and interannual variability of the high resolution GCM.

In terms of optimization, five major avenues will be investigated; semi-implicit time differencing, asynchronous communication, single PE optimization, load balancing, and parallel/asynchronous I/O. As the results indicate, Dynamics is the bottleneck due to the small time step. Development of a semi-implicit time differencing scheme is currently underway. A successful implementation would allow the time step to be raised to perhaps 2-4 minutes for the high resolution problem. Asynchronous communication will also be of most benefit in Dynamics. The SHMEM command to send data to another processor returns before the data arrives at the target PE on the Cray T3E. This extra time could be exploited to overlap communication and computation.

Single PE optimization will largely be achieved by better cache re-use. Preliminary analysis shows that the local storage for one sounding in the longwave code for the high resolution case could fit entirely in cache. Obtaining such a fit would greatly enhance performance. A similar strategy could be applied to the shortwave and Fast Physics codes. For Dynamics, if asynchronous communication is successfully implemented, it will be possible to merge some of the level loops currently separated by communication calls. This would pave the way for better cache re-use. If further single PE optimization is needed, more draconian measures such as re-organizing data structures and writing key components in assembly language will be considered. Of course, such modifications will degrade vector performance on parallel vector machines as well as the clarity of the code itself.

Off-line experimentation suggests that load balancing will improve the performance of the shortwave and LSM calculations. The re-distribution of data is determined ahead of time so the only cost is the actual communication. The benefits gained from a load-balanced polar filter would be modest. At 512 PEs, only 1/3 of the polar filter time is spent doing the actual FFT so the impact of the imbalance is small. For RAS, it is possible no improvement at all will be achieved since a great deal of the run-time would have to spent determining how the data should be re-distributed.

As mentioned, no model output was done for the results presented here. For production runs, I/O efficiency will be a factor. I/O optimization is discussed in the companion paper (Sawyer, et al., submitted to HPC '98).

In summary, this parallel GCM successfully exploits the power of MPP's such as the Cray T3E to some degree. With good scaling out to 512 PEs and reasonable single PE performance, it can integrate a $\frac{1}{2}^\circ$ by $\frac{1}{2}^\circ$ model at a rate of 5 minutes per simulated day. However, much improvement can be made, particularly in single PE performance. Using the optimization strategies mentioned above, it is hoped the run-time can be reduced to 1 $\frac{1}{2}$

minutes or less.

ACKNOWLEDGEMENTS

This project is supported by the Global Modeling and Analysis Program in NASA's Office of Mission To Planet Earth under RTOP No. 622-24-47. Access to the Cray T3E-600 was provided by the Earth and Space Sciences (ESS) component of the NASA High Performance Computing and Communications (HPCC) Program. We would like to acknowledge Jim Abeles of SGI/Cray for the help he has provided over the years in design and optimization. Thanks also go to Tom Head of Carnegie Mellon for early work on the project and to Paul Schopf of George Mason University for his ideas on the GCM design.

REFERENCES

Arakawa, A. and Suárez, M.J., 1983: Vertical differencing of the primitive equations in sigma coordinates. *Mon. Wea. Rev.*, **111**, 34-45.

Asselin, R., 1972: Frequency filter for time integrations. *Mon. Wea. Rev.*, **100**, 487-490.

Brown, J.A. and Campana, K., 1978: An economical time-differencing system for numerical weather prediction., *Mon. Wea. Rev.*, **106**, 1125-1136.

Bryan, F.O., Kauffman, B.G., Large, W.G., Gent, P.R., 1996: The NCAR CSM Flux Coupler. *NCAR Technical Note* (NCAR/TN-424+STR, May 1996).

Chou, M.-D., 1992: A solar radiation model for use in climate studies. *J. Atmos. Sci.*, **49**, 762-772.

Chou, M.-D. and Suárez, M.J., 1994: An efficient thermal infrared radiation parameterization for use in general circulation models. *NASA Technical Memorandum*, **3**, 104606, 84pp.

Déqué, M. and Piedelievre, J. Ph., 1995: High Resolution climate simulation over Europe. *Climate Dynamics*, **11**, 321-339.

Drake, J., Foster, I., Michalakes, J., Toonen, B., and Worley, P., 1995: Design and performance of a scalable parallel community climate model. *Parallel Computing*, **21**, 1571-1591.

Jones, P.W., Kerr, C.L., Hemler, R.S., 1995: Practical considerations in development of a parallel SKYHI general circulation model. *Parallel Computing*, **21**, 1677-1694.

Koster, R.D. and Suárez, M.J., 1992: Modeling the land surface boundary in climate models as a composite of independent vegetation stands. *J. Geophy. Res.*, **97**, 2697-2715.

Lal, M., Cubasch, U., Perlwitz, J., and Waszkewitz, J., 1997: Simulation of the Indian Monsoon Climatology in ECHAM3 Climate Model: Sensitivity to Horizontal Resolution. *Intl. J. Climat.*, **17**, 847-858.

Lou, J. and Farrara, J., 1996: Performance Analysis and Optimization on the UCLA Parallel Atmospheric General Circulation Model Code. In *Proceedings Supercomputing '96*, Pittsburgh, PA, USA, ACM-IEEE.

Louis, J., Tiedke, M., and Geleyn, J., 1982: A short history of the PBL parameterization at ECMWF. In *ECMWF workshop on Planetary Boundary Layer Parameterization*, Reading, pp. 59-80.

Manabe, S. and Stouffer, R.J., 1994: Multiple-Century Response of a Coupled Ocean-Atmosphere Model to Increase of Atmospheric Carbon Dioxide. *J. Climate*, **7**, 5-23.

Moorthi, S. and Suárez, M.J., 1992: Relaxed Arakawa-Schubert: A parameterization of moist convection for general circulation models. *Mon. Wea. Rev.*, **120**, 978-1002.

Shapiro, R., 1970: Smoothing, filtering and boundary effects. *Rev. Geophys. Space Phys.*, **8**, 359-387.

Simmons, A.J., Burridge, D.M., Jarraud, M., Girard, C., and Wergen, W., 1989: The ECMWF Medium-Range Prediction Models Development of the Numerical Formulations and the Impact of Increased Resolution. *Meteorol. Atmos. Phys.*, **40**, 28-60.

Suárez, M. J. and Takacs, L.L., 1995: Documentation of the Aries/GEOS dynamical core Version 2, NASA Technical Memorandum, **5**, 104606, 58pp.

Zhou, J., Sud, Y.C., and Lau, K.-M., 1996: Impact of orographically induced gravity-wave drag in the GLA GCM. *Quart. J. Roy. Meteor. Soc.*, **122**, 903-927.